OS Structures

Monolithic Systems

- **Advantages**:
  + well understood
  + easy access to all system data (all shared)
  + low module interaction cost (procedure call)
  + extensible via interface definitions
- **Disadvantages**:
  - no protection between system and application
  - not stable/robust

Layered Systems

- **Principle**: system is divided into many *layers*:
  - *each layer* uses functions and services of lower levels
  - *bottom layer*: hardware
  - *top layer*: user interface
  - *lower layers*: implement mechanisms
  - *higher layers*: implement policies (mostly)
- **Advantages**:
  + *modular*: each layer can be tested/verifies independently
  + *correctness* of layer n only depends on layer $n-1$ → simple debugging/maintenance
- **Disadvantages**:
  - just unidirectional protection
  - mutual dependencies prevent strict layering

Monolithic Kernels

- **Advantages**:
  + well understood
  + performance OK
  + sufficient protection between applications
  + extensible via definitions + static/loadable modules
- **Disadvantages**:
  - no protection between kernel components
  - side-effects by undocumented interfaces
  - complexity due to high degree of interdependency

Micro-Kernels

- **Advantages**:
  + easier to test/prove/modify
  + improved robustness/security
  + improved maintainability
  + coexistence of several APIs
  + natural extensibility
- **Disadvantages**:
  - additional decomposing
  - low performance due to communication overhead

Virtual Machines

- **Principle**: takes layered approach to logical conclusion — treats hardware + OS kernel as like they were hardware
- VM provides *identical* interface to underlying bare hardware
- OS host creates illusion that process has own processor, memory,…
- each guest gets (virtual) copy of underlying computer
- **Benefits**:
  - multiple execution environments can share same hardware
  - protection
  - controllable file sharing
  - use networking to communicate with each other
  - useful for development/testing